Training: Python Academy
High-Performance Computation with Python

TRAINING GOALS:

The requirement for extensive and challenging computations is far older than the computing industry. Today, software is a mere means to a specific end, it needs to be newly developed or adapted in countless places in order to achieve the special goals of the users and to run their specific calculations efficiently. Nothing is of more avail to this need than a programming language that is easy for users to learn and that enables them to actively follow and form the realisation of their requirements.

Python is this programming language. It is easy to learn, to use and to read. The language makes it easy to write maintainable code and to use it as a basis for communication with other people. Moreover, it makes it easy to optimise and specialise this code in order to use it for challenging and time critical computations. This is the main subject of this course.

CONSPECT:

- Optimizing Python programs
  - Guidelines for optimization.
  - Optimization strategies - Pystone benchmarking concept, CPU usage profiling with cProfile, memory measuring with Guppy_PE Framework. Participants are encouraged to bring their own programs for profiling to the course.
  - Algorithms and anti-patterns - examples of algorithms that are especially slow or fast in Python.
  - The right data structure - comparation of built-in data structures: lists, sets, deque and defaulddict - big-O notation will be exemplified.
  - Caching - deterministic and non-deterministic look on caching and developing decorates for these purposes.
  - The example - we will use a computionally demanding example and implement it first in pure Python. Then we look at some algorithmic improvements to speed up the computation.
  - Testing speed - solution to measuring how fast a program really run in Python.
  - Psyco - 'just-in-time-complier' (JIT), allowing to translate parts of the byte code to machine code. Example are used to show different possibilities of using Psyco.
  - Numerical calculations with Numpy - basic possibilities of NumPy covered.
  - Using multiple CPUS with Pyprocessing/multiprocessing.
  - Combination of optimization strategies.

Compendium Centrum Edukacyjne Spółka z o.o.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70
BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007

- Overwiew of extensions to Python with other languages.
- Python Extensions with Other Languages
  - Introduction to example that will be used in further part of that module.
  - Use of Python's C-API - Standard Python is implemented in C and offers a comprehensive API for writing extensions. The basics of this API are taught.
  - Python extensions with Pyrex/Cython.
  - Use of DLLs with ctypes - package ctypes allows to acces DLLs or shared libraries from Python.
  - Automatic generation of extensions with SWIG - The "Simplified Wrapper and Interface Generator" allows to make C/C++ libraries accesible from 13 different languages - one of them is Python. Examples in C as well as in C++ are provided.
  - Jython - basics of Python in Java implementation. Examples for use of existing Java classes as well as self writen classes.
  - IronPython - implementation of Python in .NET allowing access to all .NET features and making it a first class .NET language right next to C# and Visual Basic.
  - Use of FORTRAN subroutines from Python - example of usage F2PY to connect FORTRAN77 as well as FORTRAN90/95 programs with Python. Object-oriented interfaces to FORTRAN libraries.
- Fast Code with the Cython Compiler
  - Using pyximport to quickly (re-)build extension modules.
  - Using cython.inline() to compile code at runtime.
  - Building extension modules with distutils.
  - Fast access to Python's builtin types.
  - Fast looping over Python iterables and C types.
  - String processing.
  - Fast arithmetic.
  - Incrementally optimizing Cython code.
  - Multi-threading outsife od the GIL(Global Interpreter Lock).
  - Calling into external C libraries.
  - Building against C libraries.
  - Writing Python wrapper APIs.
  - Calling C functions across extension module boundaries.
- Numerical calculations with NumPy
  - Standard arrays and linear algebra library
  - Array-constructions and array-properties in examples
  - Speed comparison between dynamically determined Python data types with explicitly specified NumPy arrays
  - Correspondence between NumPy and C data types
  - Slicing and Broadcasting - reading and writing to arbitrary parts of arrays, applying

Compendium Centrum Edukacyjne Spółka z o.o.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70
BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007

broadcasting for arrays with different shapes.
    - Universal Functions - applying many operations on whole arrays independent from their dimensions, use examples.
    - Numerical algebra.
    - Working with missing values - masked and NA-masked arrays to handle arrays with missing or not valid values.
    - Customizing error handling - NumPy offers a fine-grained approach to handle errors without impacting the performance.
    - Testing support - NumPy includes helpers to write to test code - course introduces to testing basics with it.
- Fast NumPy Processing with Cython
    - Use of Python's buffer interface from Cython code.
    - Directly accessing data buffers of other Python extensions.
    - Retrieving meta data about the buffer layout.
    - Setting up efficient memory views on external buffers.
    - Implementing fast Cython loops over NumPy arrays.
    - Implementing a simple image processing algorithm
    - Looping over NumPy exported buffers.
    - Using "fused types" (simple templating) to implement an algorithm once and run it efficiently on different C data types.
    - Use of parallel loops to make use of multiple processing cores.
    - Building modules with OpenMP.
    - Processing data in parallel.
    - Speeding up an existing loop using OpenMP threads.

REQUIREMENTS:

- Having experience in **Python** programming.
- Basic understanding of C language is helpful - but not required.

Difficulty level



CERTIFICATE:

The participants will obtain certificates signed by Python Academy.

Compendium Centrum Edukacyjne Spółka z o.o.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70
BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007

TRAINER:

Authorized Python Academy Trainer.

Compendium Centrum Edukacyjne Spółka z o.o.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70

BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007